

**Blood Pressure Monitor Desktop Application**

**Software Workshop Team Project**

Group Raleigh

Abdullah ALI 1848984 [axa1303@student.bham.ac.uk](mailto:axa1303@student.bham.ac.uk)

Shukri ALI 1083327 [saa727@student.bham.ac.uk](mailto:saa727@student.bham.ac.uk)

Thomas CUNNINGHAM [txc754@student.bham.ac.uk](mailto:txc754@student.bham.ac.uk)

Xumin DING 1854515 [xxd715@student.bham.ac.uk](mailto:xxd715@student.bham.ac.uk)

Xinyi SUN 1786228 [xxs728@student.bham.ac.uk](mailto:xxs728@student.bham.ac.uk)

19th March 2018

School of Computer Science

University of Birmingham

Birmingham, B15 2TT

**Abstract**

Blah, blah,blah…

**Statement of Contribution**

All team members have agreed with the following contribution on the software workshop team project that they worked on with:

Abdullah ALI: %

Shukri ALI: %

Thomas CUNNINGHAM: %

Xumin DING: %

Xinyi SUN: %

Signatures: Abdullah ALI, Shukri ALI, Thomas CUNNINGHAM, Xumin Ding, Xinyi SUN

**Table of Contents**

[1. Overview 2](#_Toc509164250)

[1.1 Introduction 2](#_Toc509164251)

[1.2 Background Research 3](#_Toc509164252)

[1.3 Requirements 6](#_Toc509164253)

[1.4 Prototype 8](#_Toc509164254)

[2. System Design 11](#_Toc509164255)

[2.1 Whole system 11](#_Toc509164256)

[2.2 Database 12](#_Toc509164257)

[2.3 Network 13](#_Toc509164258)

[2.5 Graphical User Interface (GUI) 14](#_Toc509164259)

[3. Testing 16](#_Toc509164260)

[3.1 Functional Tests 16](#_Toc509164261)

[3.2 Junit Tests 18](#_Toc509164262)

[4. Team Organisation 19](#_Toc509164263)

[5. Evaluation 19](#_Toc509164264)

[6. Appendix 20](#_Toc509164265)

# 1. Overview

# Introduction

An innovative medical application that connects the doctors to their patients and enables doctors to receive effortlessly the regular patience blood pressure readings. Currently, the way the blood pressure reading is obtained is either the patience goes to the clinic and the nurse does the reading or the patience records the reading on paper at home and passes on to the nurse. This is certainly a tedious process and requires staff to focus on paper work instead of more essential work.

Blood pressure monitoring is very important and for some patients can mean live or dead situation and so the more efficient it can be done the better it is for all parties involved.

We assume the patient is provided with equipment that can monitor the blood pressure. Once the equipment has completed the reading, the patient is able to obtain three blood pressure readings and register that into the app with time and date recorded.

From the patient perspective, the patient is able to login into the application with the provided unique username and password. Once logged in, the patient has number of options. The patient can submit the reading by pressing a button, the reading will then be uploaded into the database. To ensure that patient is aware of the progress made over time they can view some indicators in numbers and graph displayed in their homepage. In addition, the patience is able to find instruction to perform the blood pressure reading and if needed can watch a video of how to obtain these readings.

From the doctor perspective, once logged in the doctor can search for patients based on their name or prefix. In addition, the doctor is able to see a list of patients that have completed their readings. The doctor should be able to select a patient, view their details and if necessary export a pdf file.

# 1.2 Background Research

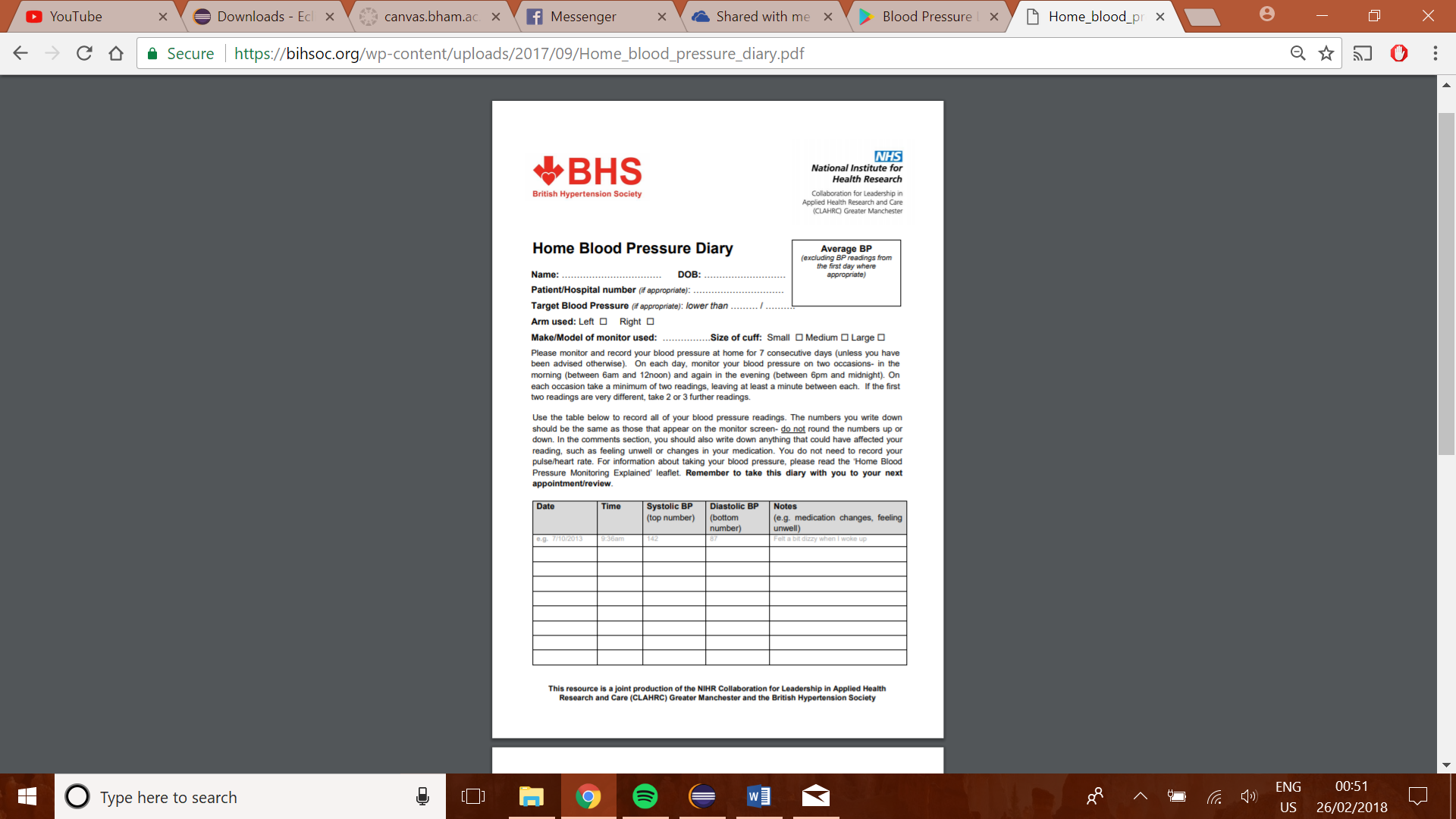
Blood pressure is represented by two values. The systolic blood pressure, which is the pressure that the blood is forced through the blood vessels during a heartbeat, and the diastolic blood pressure, which is the pressure in blood vessels between heartbeats. Hypertension, also commonly known as high blood pressure is when either one, or both numbers are abnormally high. Estimates show that around 40% of the world's population has hypertension[[1]](#footnote-1), which is an enormous amount. Hypertension is a precursor to many life-limiting conditions like heart failure and kidney failure, so it is important to screen people regularly for this condition and to manage them appropriately.

Blood pressure measurements are difficult to record accurately in a clinical setting, as the anxiety and stress of the experience tends to give inaccurately high readings[[2]](#footnote-2). Therefore, to diagnose and manage hypertension, patients are required to monitor their readings at home for a given period; the average of which is used to confirm the diagnosis of hypertension.

Current guidelines suggest that a patient should record at least 2 consecutive readings, twice a day, for ideally 7 days (a total of 28 readings)[[3]](#footnote-3). Currently, most patients are given a paper diary, on which they record their blood pressure readings. The paper is then handed to a clinician, whom then manually uses those readings to calculate the average and determine whether the patient meets the criteria for confirmation of a hypertension diagnosis. As one can imagine, this is not only wasteful of resources like paper and printing but is also wasteful of the clinician and patient's time. By digitalising this process, we can save the time taken for the patient to manually take the readings back to the clinician, and for the clinician to manually calculate the average. Not to mention eliminating the risk of the patient losing their diary.

Existing Products Review

In this section, our aim is to analyse products and services currently available, for home BP monitoring. A point to be noted is that none of the products were found to solve the main problem we identified, which was of being able to transfer the readings directly to the clinician.

Product 1: Paper Diary[[4]](#footnote-4)

This is a paper diary for home BP recording. This is created by a leading Hypertension authority and most clinicians today use this or a variation on this style of paper diary.

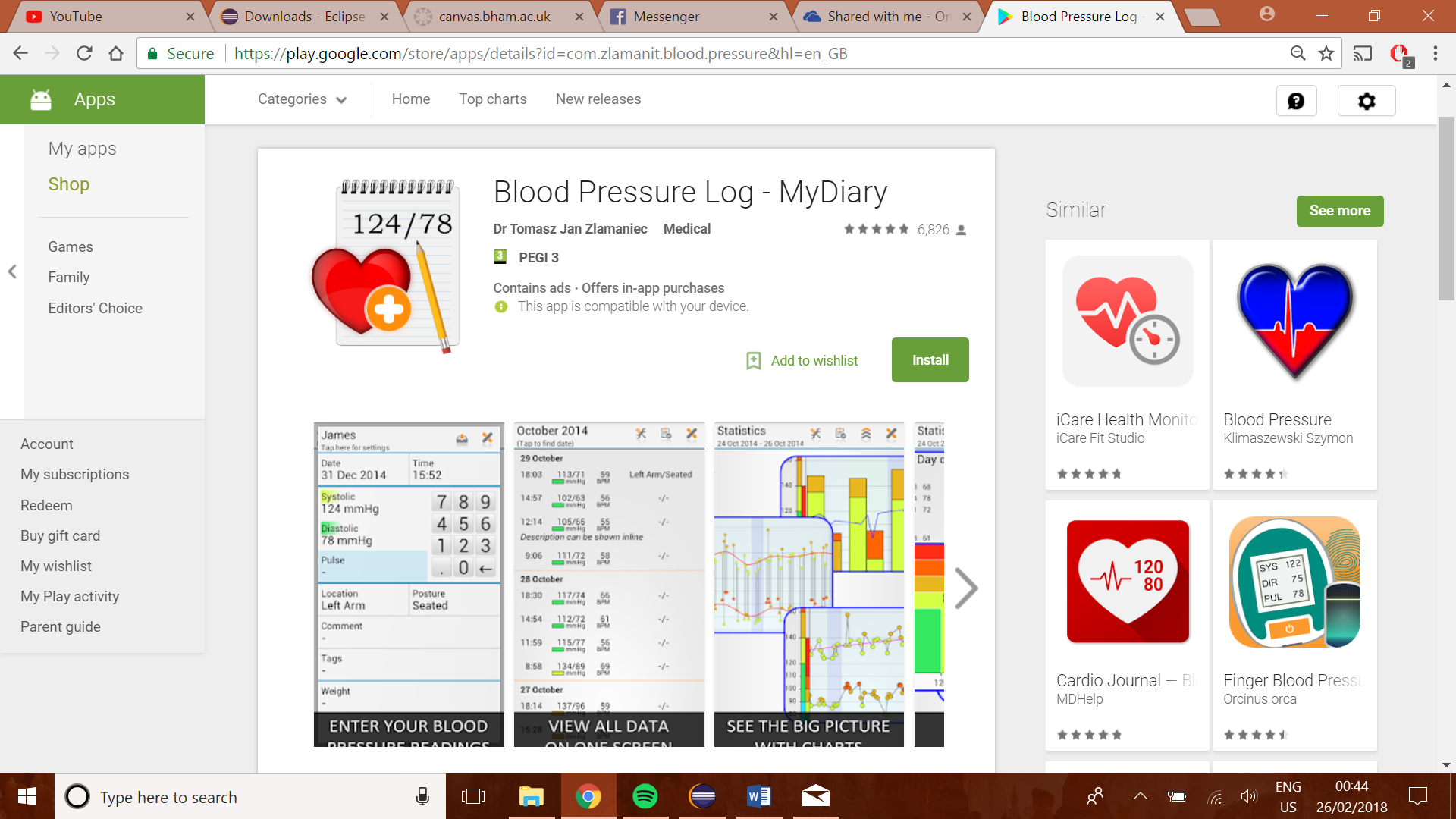
Pros:

* Instructions on how to record the readings, written clearly; by experts in the field.
* Easy to use, with clear indication of how exactly to fill in the form, with addition of a comments section for any extra comments about that reading.
* Secure (can't be hacked)

Cons:

* Requires readings to be manually inputted and the average to be manually calculated
* Is not able to show visually, the patient's blood pressure trends.

Product 2: Blood Pressure Log – MyDiary (Android App)[[5]](#footnote-5)

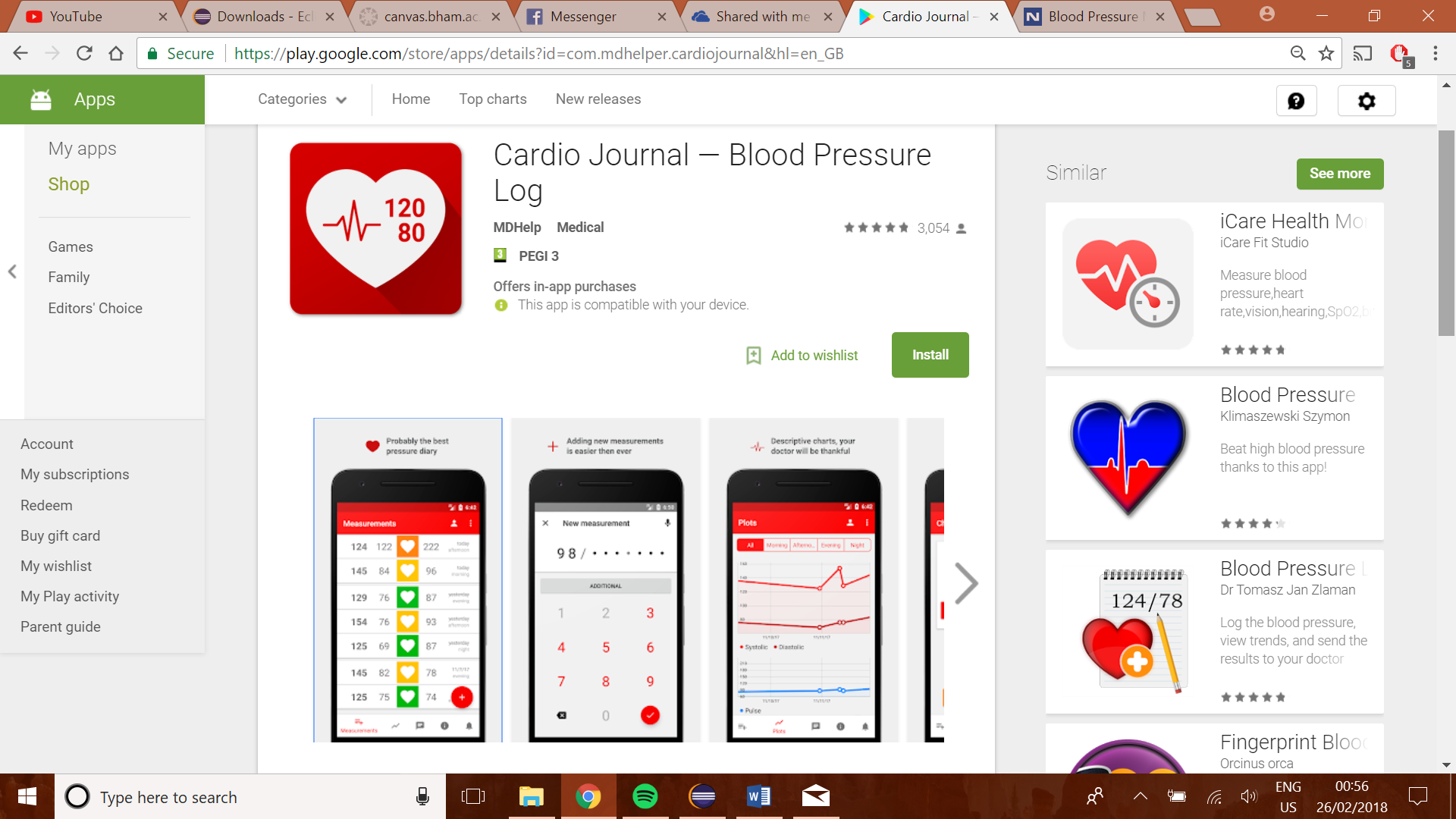
This is an android app which can be used as a diary. This allows us to record blood pressure readings, which can then be displayed numerically, as well as graphically.

Pros:

* Patient can easily enter their readings
* Readings are represented visually as graphs of trends over time
* Extra information like location, posture, weight, and comments can also be added.

Cons:

* Very cluttered interface
* No way of transferring the readings across to the clinician

Product 3: Cardio Journal – Blood Pressure Log (Android App)[[6]](#footnote-6)

Pros:

* Very simple and visually pleasant graphical user interface
* numerical display for inputting readings, to prevent entering of invalid characters or in an invalid format.

Cons:

* This app also does not provide a feature for sending the readings to the clinician automatically.
* The app provides no information on how to take readings appropriately as well as when to take them.

In summary, research into existing systems has been very informative in highlighting features which will be potentially very useful in our project. It highlighted that the unique feature of our app will be the connectivity with the clinician, so this should be our main focus. We also realised that while the app should be clear and informative, it should also have a clean GUI which is easy to use and navigate.

# 1.3 Requirements

Functional

1.3.1 User login

* For login purposes, the system shall require the user to input his/her username and correct password.
* The system shall notice the user to try again account, when they wrongly input username or password, and show “wrong username or password!”.
* The system should check if the user has signed up, if not, show “user does not exist” message.
* The system should distinguish users from doctor and patient. The doctor should log into the doctor’s home page, while the patient should log into the patient’s home page.
* The system will encrypt the user password.

1.3.2 Patient

* The app should enable the patient to input 3 blood pressure readings, with date and time it was recorded.
* The app should enable patients to write comments with their BP and upload it.
* The database must be able to store patient blood pressure readings.
* The app should record all readings of patient and present them in a graph to show a progress.
* The app should enable patients to set a target blood pressure reading, both in text and chart.
* The app should enable automatic calculation of average value of all blood pressure readings they have uploaded and present it both in text and chart.
* The app should also calculate and indicate the highesta and lowest BP recorded.
* The app should enable automatically update reading left once the patient upload his/her readings and present it in text.
* The app should update the graph instantly after the reading is submitted.

1.3.3 Doctor

* The app should allow doctors to create new patient account that contains all their information.
* The doctor should be able to set the blood pressure target for the patient.
* The database must be able to store basic patient information.
* The system should generate username and password of patients automatically when the doctor adds a new patient.
* The app should enable doctors to search a patient by name or prefix, and show their name, date of birth, and address in a patient list.
* The app should enable doctor to choose a patient showed in both lists by double click, and there should be a new window contains information (name, readings, etc.) of this specified patient.
* The app should show the patients that completed their readings in the completed list.
* The app should enable to generate PDF of patient information showed in patient view.
* The app should show in the patientView page a graph indicating patient progress.
* The should show the basic statistics of the patient bp progress in the patient view page.

Non-functional

* (Efficiency) The application shall provide a response to user input within 1 second.
* (Security) The application shall ensure the all the information (including users’ personal information, patients’ readings etc.) can only be accessed by the patients themselves and their corresponding doctors.
* (Availability) The application should be able to access at any time as long as users need.
* (Capacity) The system must enable concurrently support for multi users at a time.
* (Scalability) The system should allow as many doctors and patients to use the system.

# 1.4 Prototype

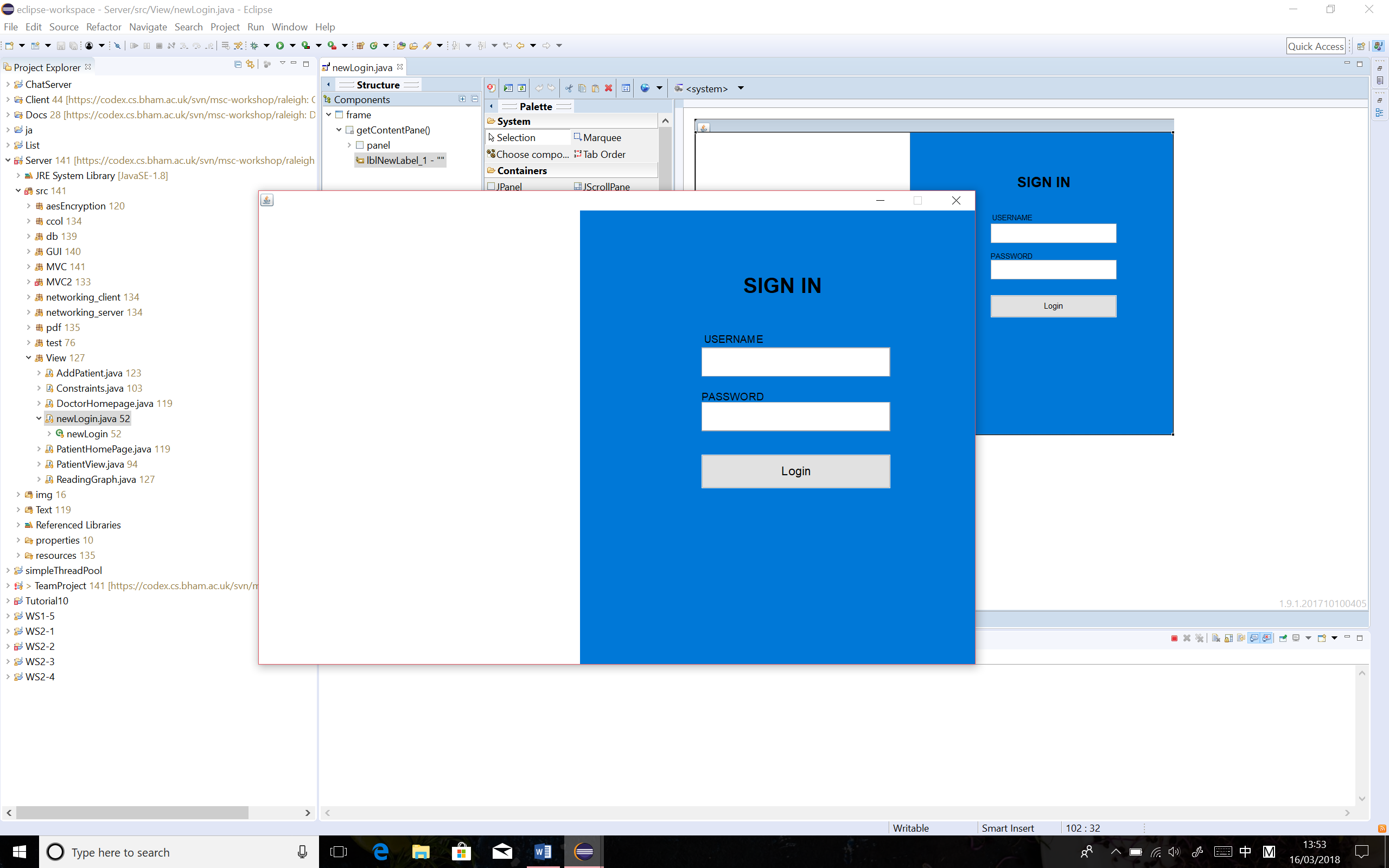
The initial prototype was built up to visualise the concept of the blood pressure monitor desktop application, based on the requirements which were proposed at the early stage of the whole project. The prototype of this application would include a user log in page that requires users, in this case would be either patients or doctors, to log in the application using their own username and password. (See Figure 1)

Figure 1 Prototype of User Login Page

Once patients/doctors log in successfully, another window will be opened to display their homepages, which varies from the type of users. The registered doctors will see the doctor homepage and the registered patients will be given the patient homepage.

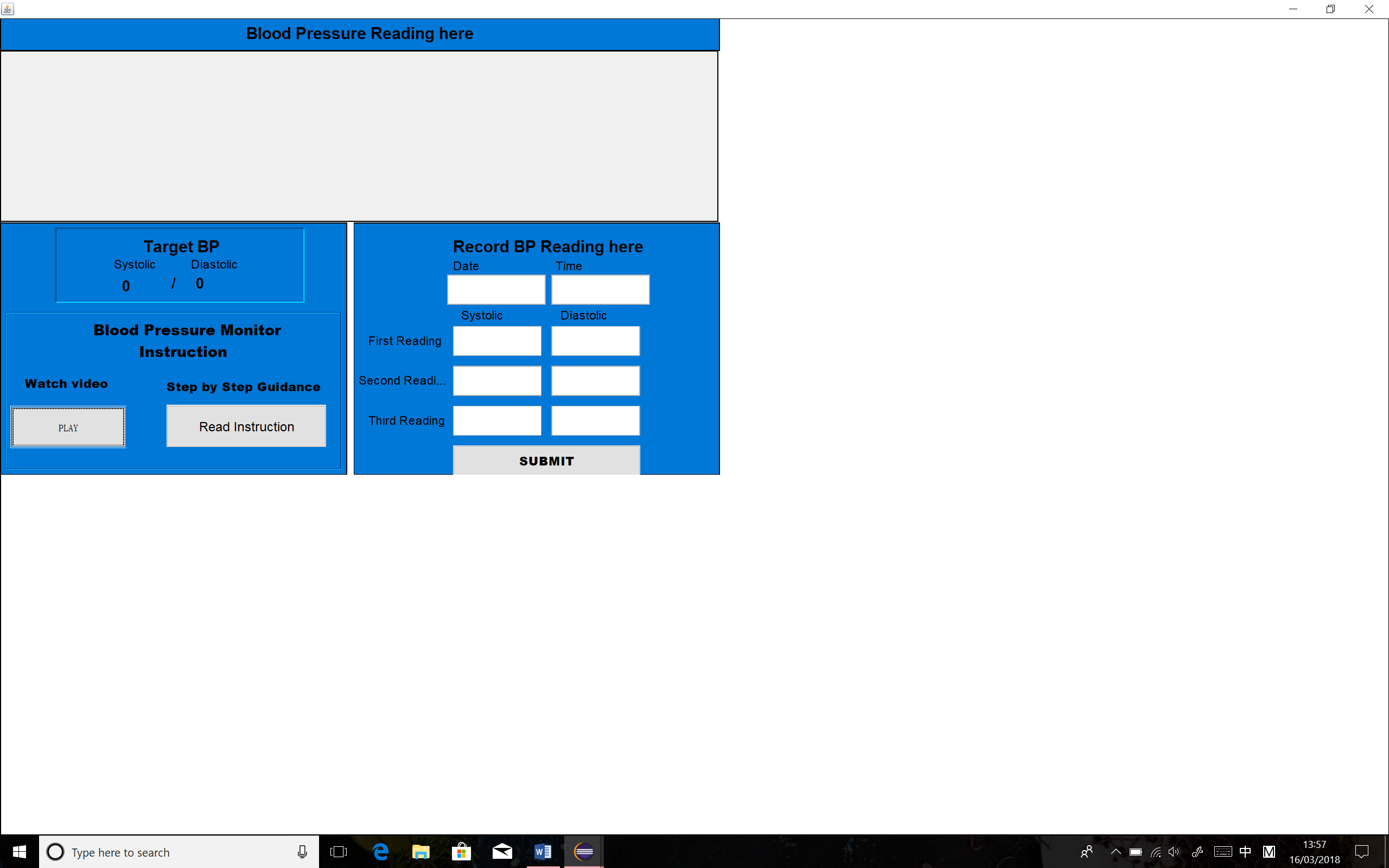
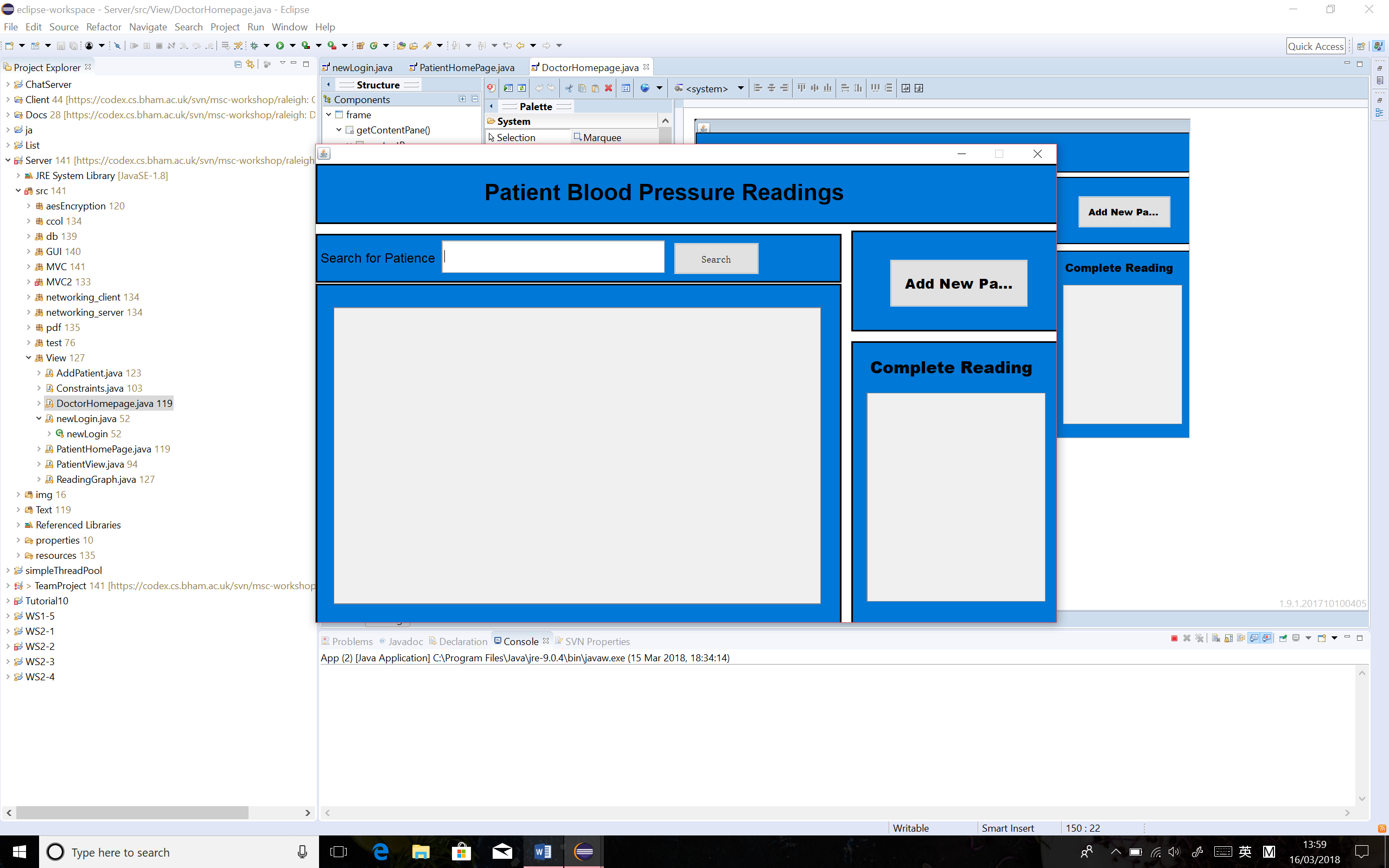
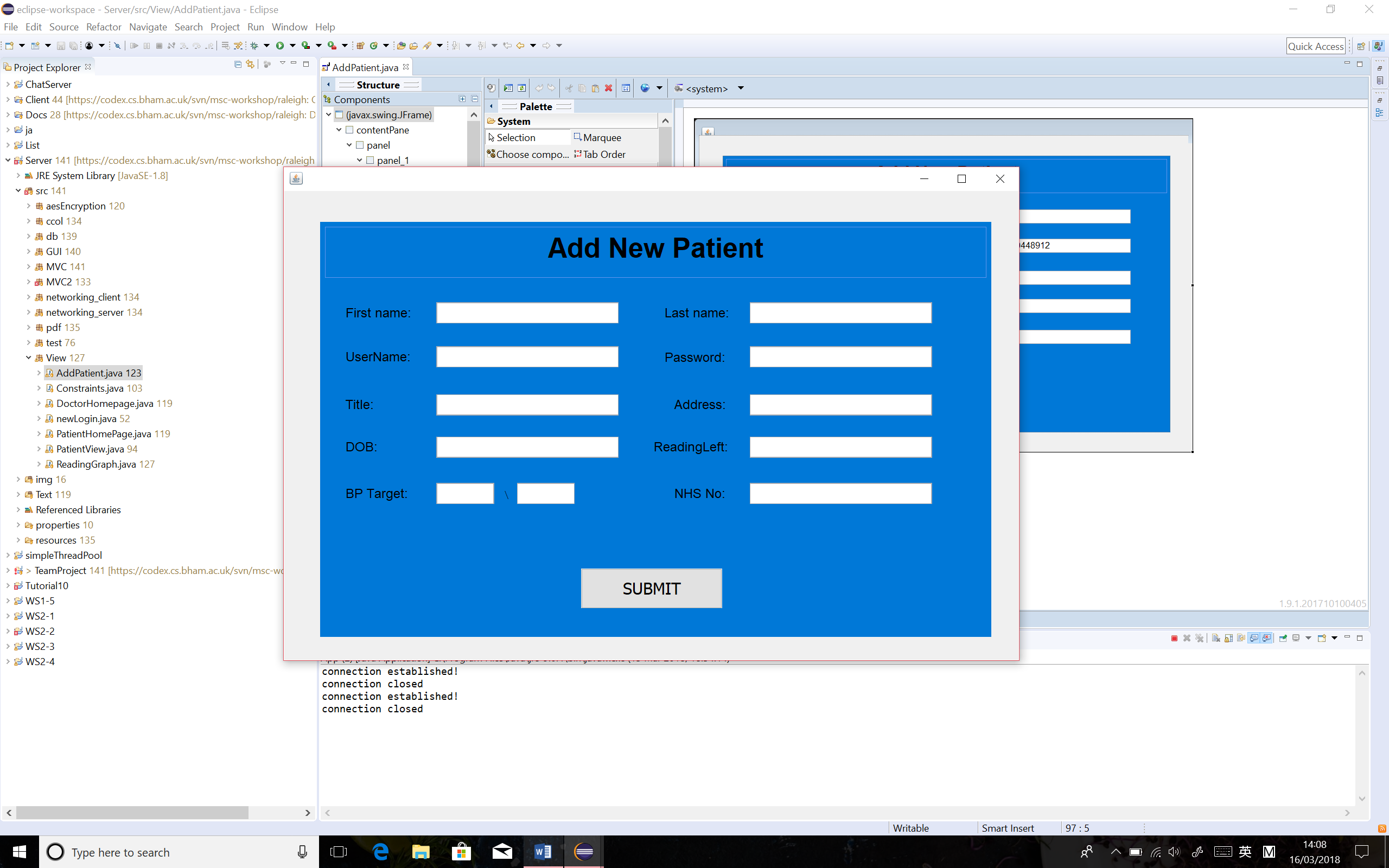
On the patient home page, a relevant video and readable guidance are provided to help patient understand how to use the blood pressure monitor. The BP target that was prescribed by the doctors was printed on the screen. After measurements, the patient allows to input and submit the blood pressure readings, along with the recorded time and date, to track the BP trend on a daily base. The BP trend graph will be presented on the top of the page as a real-time update. (See Figure 2)

Figure 2 Prototype of Patient Home Page

Figure 3 Prototype of Doctor Home Page

Doctors are permitted to access the registered patients’ details through the doctor home page. From the prototype of doctor home page, it was given the functionality to search existed patients and add new patients. To add a new patient, the doctor needs to click on the “add patient” button, which opens a new window that the doctors can input all the details required for creating a new patient. To read the patient up-to-date blood pressure readings, the doctor can click on the patient he searched, which pops up a new window that displays the most important patient information on the screen, such as patient’s personal details, blood pressure target and date of the last reading etc. The BP reading trend graph that generated on the patient home page was also plotted on this page, sending an idea to the doctor that how the patients’ blood pressure keeps changing over time. Doctors can also view a list of patients who have completed the prescribed reading amounts on the doctor home page. (See Figure 3, 4 and 5)

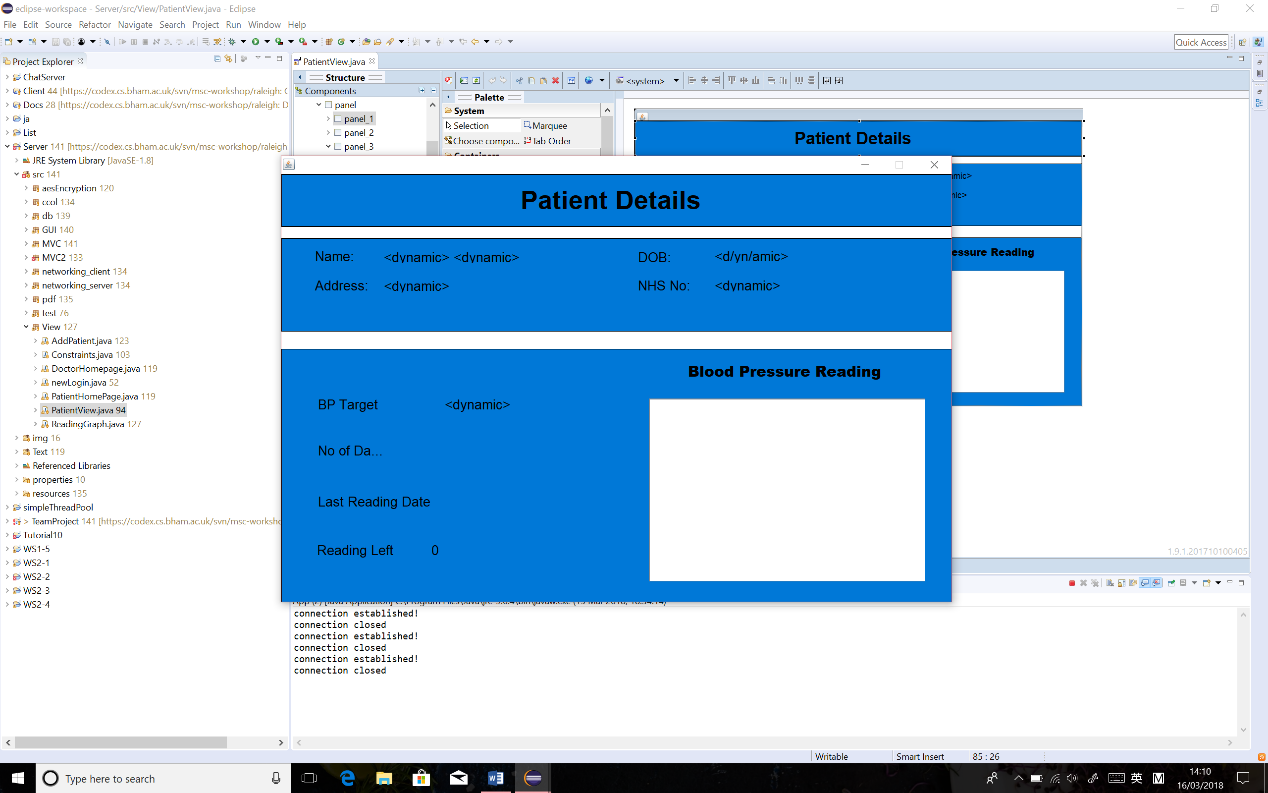
Figure 4 Prototype of Add Patient Page

Figure 5 Prototype of Patient Details Page

# 2. System Design

# 2.1 Whole system

Our blood pressure submission system exists in a three-tier architecture. This architecture is composed from a Client, a Server and a Database. The client is responsible for making user driven requests of the server, which depending on the nature of the request, it will then query the database for the required information to return to the client application.

We constructed this system in distinct parts with specified functionality and interactions between each of these parts. With these specifications agreed it was therefore possible to work concurrently providing that each of us achieved the goals that we had set out, as we could be sure that upon completion each of the parts should produce a correctly functioning program. This also aided the testing of each components. We could more easily divide functionality into smaller units and test those, as well as testing for correct interaction between components in isolation.

Typical use of the system is as follows: Any user of the system is presented with a login screen as the view of the client application. The user will then input their login details, which will identify whether the user is stored as a doctor or a patient. The password which the user types is encrypted using AES and then wrapped in a serialised “TransmissionObject” which contains and carries the login information across the network to the server application. Once it has arrived at the server, the credentials are unencrypted and used to verify the user by querying the database for those credentials. If verified, the client will be returned either a “Patient” or “Doctor” type of the abstract “User” class, otherwise a null object is returned. Based on this information, the user is presented with a new window that is relevant to either a patient or doctor. A patient is then able to submit new readings and review the ones that they have recorded so far, as well as other minor functionality such as watching instructional videos on blood pressure readings. A doctor however can review all the patients that are currently stored in the database and the readings they have submitted. They are able to search for patients by searching for their name, and are notified in real time as patients complete their readings.

The communication between client and server is via a serialised class called “TransmissionObject”. This object manages all communication between the two, where it specifies what request/information is being provided via instance variables and encapsulates objects that contain information such as subclasses of the “User” object.

Due to the nature of the information that is moved throughout the system, computation of the information is primarily handled on the server, whilst the organisation of how it is to be presented is handled within the model. This ensures that information is stored and handled efficiently whilst contained within the database, but also that when the appropriate information is apprehended at the client it is presented to the user of the system in the most suitable and accessible way possible.

# 2.2 Database

While deciding on how the database should be constructed, a significant decision was regarding whether we should be storing User data as whole objects, or as individual data. We decided to choose the latter as this allowed us to be able to manipulate specific user fields and query certain users via SQL, as opposed to loading the objects to memory, making the changes in JAVA and then storing the object back into the database. This allowed our system to be able to query data very quickly.

Our database contains two tables. A users table, which stores all the users which have been created, as well as a readings table, which stores all the readings which have been submitted. A common user\_id field, which is present in both tables is then used to determine which reading belongs to which user.

Another consideration was whether to add the constraints at the database level for invalid data entry. We eventually decided that it is best for invalid data entry to be stopped at the point of entry in the client GUI, rather than sending it to the server and waiting for the server to respond with an error. Therefore, the database class methods require certain pre-conditions regarding the validity of the parameters, when data is being entered or retrieved from the database.

* How does the JDBC work? ie, how does a login request in java get converted to an SQL query?
* Why did you use prepared statements and not statements?You do not need to put code in this section, but you might like to screenshot a method and talk about it, if you feel it helps you to explain your implementation.
* Entity relationship diagram -- how have you normalised the database? why is this important? Why have you chosen to use a relational database.

# 2.3 Network

Parts of the server and client sides with protocols

# 2.5 Graphical User Interface (GUI)

The GUI is one of the crucial parts of the application and without it, it would be near impossible to have the patient and doctor interact.

Based on the prototype we had at the start of the project, there were few function that the GUI had to include in order to facilitate the necessary communication between the patient and doctor. So initially we wanted a login page for both the patient and doctor, a doctor homepage and patient homepage which would include all the components for each user. The doctor homepage would include search for patients, view their details, a list of the patient who completed the reading and a function to add new patients. The patient page had to include a function to input their reading which consists of three blood pressure readings, date/time and comments, the ability to see their progress in their blood pressure levels and information if they needed help with their blood pressure reading.

We later realised that having all of these components will make the pages crowded and it would benefit if we can have more pages to increase the readability of the page. Therefore, we have added separate pages for adding patients and viewing patients details.

As we are implementing the three-tier multilayer architecture, the GUI is the part that initiated and completes the requests of the application process. It is located on the client side and to ensure the maintainability of the application we have separated the model from view. To do this we have constructed a class called model which performs like a link between the business logic and the view classes. This will ensure that changes done to the view classes will not change the working of the logic and databases. This way, the view requests and receives data but will never influence the structure of the business logic of the application.

Patient and doctor both have the same login page and once they type in their username and password and press the login button then action is performed to check whether those details are correct, if not then a popup message appears stating that those details are incorrect. If they are correct the login page will be disposed and then the model will generate the correct user object to allowing the appropiate homepage to appear.

In the doctor home page, when the doctor searches for a patient details, we use the JList to show the result in a ScrollPane. After the doctor types the name or a prefix and presses the search button which activated the actionPerformed, we implement the setViewPortView method to show the result in the ScrollPane. In order for the doctor to view the selected patient details we have addMouseListener to allow for double click on the patient name then to display ViewPatient page with the corresponding patient details. The ViewPatient page mostly exists of JLabels that show the patient details which are mainly obtained through the patient object getters, except the blood pressure indicators which are obtained through the methods from the model class that calculate the numbers such as the overall average. In addition it shows the same line graph that we had on the patient home page. Additionally, when the doctor a new patient, the new page appears to create the patient account. We have placed number of constraints to ensure that the details put in by the doctor are aligned with the expected values. Once the submit button is pressed the details are added to the database, message appears that it is successfully submitted and then goes back to the doctor homepage.

In the patient home page, to keep the patient informed we have placed a line graph that shows the blood pressure progress over time with the additional line of the target blood pressure making the patient constant aware of it. Also this will be in real time, when the patient submits their readings it will automatically update the graph. As java swing graph components are limited, we have decided to take the JavaFX graph component and attach it in a swing panel. We did this by using the JFXPanel, which allows us to add a JavaFX item into our GUI. We have a separate class named ReadingGraph that maintains the method initFX that implements the JFX component into java swing and another method that plots the blood pressure reading for the patient on to a XY graph.

In addition the page also displays the numbers of the overall average blood pressure, the target blood pressure and the number of readings that are left to be submitted by the patient. These information is obtained by requesting the details from the specific patient object and we use the getters in the patient class.

To add the readings, the patient types in the date/time, 3 blood pressures and comments, once the button is pressed it will call the actionPerformed method and perform a task. This then will call number of methods from the class constrains to check whether all the text field are filled and follow the constrains and if not, the patient will see a message asking to try again. If the constraints are satisfied, then it calls the addReading method in the model class from the Communicate class which is a networking wrapper for adding a reading to the databases. Then an update method is called that ensures that all methods that call for these data in the model class are notified.

# 3. Testing

# 3.1 Functional Tests

|  |  |  |
| --- | --- | --- |
| **Test** | **Action** | **Sent requests and expected Response** |
| 1 | On the log in page, click the “login” button with correct username and password. | A user request is sent to server to check the database; the input matches and a response is sent to login into the either a patient home page or a doctor home page. |
| 2 | On the log in page, click the “login” button with incorrect username or password. | A user request is sent to server to check the database; the input does not match, and a response is sent to inform the user that the login has failed with a reason explained in the pop-up window. |
| 3 | On the patient home page, click the “play” button. | A patient request is sent to server, a response is sent back link to the web site where the video is stored. |
| 4 | On the patient home page, click the “Read Instruction” button. | A patient request is sent to server, a response is sent back link to the web site where the instruction is stored. |
| 5 | On the patient home page, select a future date and time that recorded the blood pressure, along with correct inputs in the blood pressure reading fields and click “submit” button. | A patient request is sent to server, a response is sent to warn the patient the submit failed, with a reason of inappropriate date inputs, explained on the pop-up window. |
| 6 | On the patient home page, type non-numerical inputs into the blood pressure readings fields and click “submit” button. | A patient request is sent to server, a response is sent to warn the patient the submit failed, with a reason of wrong format of inputs, explained on the pop-up window. |
| 7 | On the patient home page, type numerical inputs into the blood pressure readings fields, with some empty fields left and click “submit” button. | A patient request is sent to server, a response is sent to warn the patient the submit failed, with a reason of “the field cannot be empty”, explained on the pop-up window. |
| 8 | On the patient home page, type numerical inputs into all blood pressure readings fields, where the systolic is over 300 or less 30 and click “submit” button. | A patient request is sent to server, a response is sent to warn the patient the submit failed, with a reason of wrong inputs, explained on the pop-up window. |
| 9 | On the patient home page, type numerical inputs into all blood pressure readings fields, where the diastolic is over 200 or less 20 and click “submit” button. | A patient request is sent to server, a response is sent to warn the patient the submit failed, with a reason of wrong inputs, explained on the pop-up window. |
| 10 | On the patient home page, input all the blood pressure readings to the corresponding textbox in correct format, along with the recorded date, time and any comments(optional), click “submit” button. | A patient request is sent to server, a response is sent back a “submission success” message in the pop-up window and plot the inputted blood pressure readings on the graph accurately. A latest average blood pressure reading is computed. The screen prints out remaining blood pressure readings for the patient. |
| 11 | When the remaining BP reading says “1” left, input the last readings data into all the fields and click “submit” button. | A patient request is sent to server, a response is sent back a “you have completed all your readings” message in the pop-up window and plot the inputted blood pressure readings on the graph accurately. A latest average blood pressure reading is computed. The screen prints out “0” remaining blood pressure readings for the patient. |
| 12 | When the remaining BP reading says “0” left, input the last readings data into all the fields and click “submit” button. | A patient request is sent to server, a response is sent back a “you already finished your readings” message in the pop-up window. |
| 13 | On the doctor home page, search the patient by typing in an unregistered patient name in the search bar, click “search” button. | A doctor request is sent to server, a response is sent to inform the doctor that the inputted name does not exist in the database in a pop-up window. |
| 14 | On the doctor home page, search the patient by typing in a registered patient name in the search bar, click “search” button. | A doctor request is sent to server, a response is sent to list all the possible search results in the textbox below. |
| 15 | Double click on a registered patient displayed in the result list that the doctor is looking for. | A doctor request is sent to server, a response is sent to open the new window – view patient page, which presents the chosen patient’s personal information collected from the database, including name, DOB, address and NHS number. Each BP reading that patient has entered is listed in the right bottom box in detail. The target, average BP, highest BP, lowest BP and remaining BP readings will be computed and displayed on the left bottom. An up-to-date patient BP trend graph is plotted in the middle of the page. |
| 16 | On the view patient page, click “generate PDF” button. |  |
| 17 | On the doctor home page, click the “add patient” button. | A doctor request is sent to server, a response is sent to open add new patient page in a new window. |
| 18 | On the add patient page, input the information with some empty fields left and click “submit” button. | A doctor request is sent to server, a response is sent to warn the doctor that the submission is failed, with a reason of “the field cannot be empty”, explained on the pop-up window. |
| 19 | On the add patient page, input the first name of the new patient in illegal format whilst all the other information is correctly entered, click “submit” button. | A doctor request is sent to server, a response is sent to warn the doctor that the submission is failed, with a reason of “the first name must be between 1 to 30 characters, explained on the pop-up window. |
| 20 | On the add patient page, input the last name of the new patient in illegal format whilst all the other information is correctly entered, click “submit” button. | A doctor request is sent to server, a response is sent to warn the doctor that the submission is failed, with a reason of “the last name must be between 1 to 30 characters”, explained on the pop-up window. |
| 21 | On the add patient page, input the number of readings left as “0” whilst all the other information is correctly entered, click “submit” button. | A doctor request is sent to server, a response is sent to warn the doctor that the submission is failed, with a reason of “the number of readings left must be between 1 to 30”, explained on the pop-up window. |
| 22 | On the add patient page, input the systolic of BP target as “400” whilst all the other information is correctly entered, click “submit” button. | A doctor request is sent to server, a response is sent to warn the doctor that the submission is failed, with a reason of “the systolic must between 30 to 300”, explained on the pop-up window. |
| 23 | On the add patient page, input the diastolic of BP target as “300” whilst all the other information is correctly entered, click “submit” button. | A doctor request is sent to server, a response is sent to warn the doctor that the submission is failed, with a reason of “the systolic must between 20 to 200”, explained on the pop-up window. |
| 24 | On the add patient page, input all the patient information to each corresponding textbox in correct formats, click “submit” button. | A doctor request is sent to server, a response is sent to generate a new patient, in the database and the new patient allows to log in to the application with correct username and password. |

# 3.2 Junit Tests

Blah, blah, blah…

* Just mention that all j-unit tests passed and for readers to check the code to see these.

# 4. Team Organisation

Our project is split to briefly four parts: database, server, client, GUI, each team member took a main part as following:

Thomas: the networking between server and client.

Abdullah: database

Shukri: GUI

Xinyi: actions performed in doctor part(model).

Xumin: actions performed in patient part(model).

When combining each section, there was several modifications in client part and GUI part as following:

Abdullah: Add graph and PDF generating function in GUI.

Shukri: Separate GUI containing all actions into model-view-controller architecture.

To cover risk of team members absence, we have made sure that we shadow each other’s work. So, we have assigned Abdullah and Tom to work closely and Shukri, Xinyi and Xumin to work closely.

The report organization is as following:

* Thomas: Whole System Design
* Abudullah: Background Research/Database
* Shukri: Introduction/GUI
* Xinyi: Requirements/Team Organization
* Xumin: Prototype/Use case/Testing/Diagrams

Regular meeting twice a week on Monday and Tuesday. During the last week it became daily meeting. The meeting/project dairy can be seen in the appendix. In addition, the team constantly communicated through Facebook messenger.

# 5. Evaluation

# 6. Appendix

1. http://www.who.int/gho/ncd/risk\_factors/blood\_pressure\_prevalence\_text/en/ [↑](#footnote-ref-1)
2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1120141/ [↑](#footnote-ref-2)
3. https://www.nice.org.uk/guidance/CG127/chapter/1-Guidance#diagnosing-hypertension-2 [↑](#footnote-ref-3)
4. https://bihsoc.org/wp-content/uploads/2017/09/Home\_blood\_pressure\_diary.pdf [↑](#footnote-ref-4)
5. https://play.google.com/store/apps/details?id=com.zlamanit.blood.pressure&hl=en\_GB [↑](#footnote-ref-5)
6. https://play.google.com/store/apps/details?id=com.mdhelper.cardiojournal&hl=en\_GB [↑](#footnote-ref-6)